Universidad Tecnológica Nacional
AJEA – Actas de Jornadas y Eventos Académicos de UTN
Trabajo Completo Publicado en AJEA – Full Text Published in AJEA
https://rtyc.utn.edu.ar/index.php/ajea/article/view/1628
Libro de actas de resúmenes DOI: https://doi.org/10.33414/ajea.1640.2024
Colección JIT en AJEA https://rtyc.utn.edu.ar/index.php/ajea/JITs

Las herramientas para la mitigación de riesgos de las aplicaciones web.

The tools for risk mitigation in web applications.

Presentación: 17/10/2023

Jonatan Makianich:
Universidad Tecnológica Nacional Facultad Regional Rosario Argentina
jonatan8090@gmail.com

Yoana Lomo:
Universidad Tecnológica Nacional Facultad Regional Rosario Argentina
yoanalomo@gmail.com

Bautista Guerra:
Universidad Tecnológica Nacional Facultad Regional Rosario Argentina
bautistaguerra.it@gmail.com

Ornella Colazo:
Universidad Tecnológica Nacional Facultad Regional Rosario Argentina
ornecolazo@gmail.com

Maximiliano Mansilla:
Universidad Tecnológica Nacional Facultad Regional Rosario Argentina
mmansilla02@outlook.com

Guillermo Dolan:
Universidad Tecnológica Nacional Facultad Regional Rosario Argentina
guillermopatdolan@gmail.com

Lucía Morena Fabbri:
Universidad Tecnológica Nacional Facultad Regional Rosario Argentina
fabbriluciam@gmail.com

Resumen

El siguiente trabajo está orientado hacia los mecanismos para mitigar y prevenir las perdidas y/o robos de la información de las organizaciones en sus aplicaciones webs, a través de herramientas como el SAST, DAST, IAST y pentesting. Exponiendo al final un ejemplo de ataque en donde una empresa es víctima y la amenaza que esta conlleva.

Palabras clave: SAST, DAST, IAST, PENTESTING, CODIFICACIÓN SEGURA.

Abstract

The following work is focused on mechanisms to mitigate and prevent losses and/or theft of organizations' information in their web applications, using tools such as SAST, DAST, IAST and pentesting. It concludes by presenting an example of an attack in which a company falls victim and the threat it entails.

Keywords: SAST, DAST, IAST, PENTESTING, SECURE CODING.

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes DOI: https://doi.org/10.33414/ajea.1640.2024
Texto Completo Publicado en AJEA – Full Text Published in AJEA
https://rtyc.utn.edu.ar/index.php/ajea/article/view/1628

Introducción

Con el creciente aumento de los sistemas de información, poco a poco ha empezado a tener más importancia la seguridad y el resguardo de la información dentro de los sistemas informáticos, lo que antes era visto como algo adicional, ahora es considerado un requerimiento funcional dentro del código. Esto se debe a que la información paso a ser uno de los activos más importantes de cada organización, por lo cual no es sorpresivo que la seguridad del código comience a ser un elemento fundamental. Los estándares de seguridad se pueden utilizar como guía o marco para desarrollar y mantener un sistema de gestión de seguridad de la información (SGSI) adecuado. Las normas ISO/IEC 27000, 27001 y 27002 son normas reconocidas mundialmente. Con la certificación de estas normas las empresas pueden tener su SGSI certificado por una organización externa y garantizar tener resguardos de la seguridad informacional. Dentro de los nuevos controles de la ISO 27002:2022 podemos encontrar a la codificación segura en la sección 8.28.

Codificación Segura

Los siete puntos claves en el desarrollo de un software seguro, como fueron establecidos por el Instituto de Ingeniería Eléctrica y Electrónica (IEEE) y popularizados por Gary McGraw y Cigital, sugieren que se deben considerar los siguientes aspectos (McGraw, 2004: 80-83):

- Construcción de casos de abuso. La creación de casos de abusos trata de definir de manera explícita cómo y qué aspectos deben ser resguardados, frente a qué tipos de amenazas o entidades, y durante que período de tiempo.
- Requerimientos de seguridad. En los requerimientos funcionales deben encontrarse requerimientos de seguridad.
- Análisis de riesgos. Dentro del ámbito de diseño y arquitectura, debe implementarse un proceso de análisis de riesgos y seguridad para otorgarle a proyecto normas para el comienzo de la construcción del mismo.
- Revisión externa. Es una evaluación imparcial y objetiva llevada a cabo por sujetos ajenos al proyecto.
- Pruebas de seguridad basadas en los riesgos. Estas evaluaciones deben abarcar tanto pruebas de seguridad funcional utilizando métodos de prueba estándar como pruebas de seguridad basadas en el riesgo utilizando patrones de ataque como referencia.
- Revisión de código. En el proceso de desarrollo de software, se genera al menos un código fuente. En el nivel del código, el enfoque se dirige hacia la detección de errores en tiempo de ejecución y la identificación de vulnerabilidades que podrían comprometer el progreso del proyecto. Sin embargo, para lograr un software resistente a amenazas, es esencial llevar a cabo todas las etapas del modelo de desarrollo.
- Pruebas de penetración. A través del hacking ético controlado, se realizan pruebas con el fin de identificar inconvenientes que se pudieran presentar en el futuro, para evitar y prevenir incidentes.
- Operaciones de seguridad. El monitoreo del comportamiento de seguridad del software es una técnica defensiva fundamental.

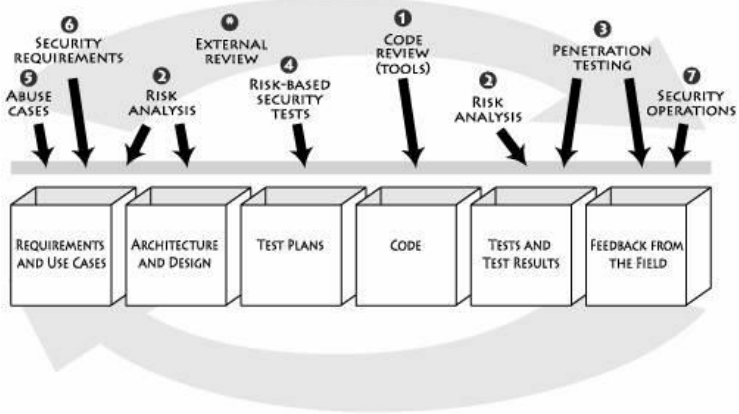

Figura 1 – Puntos claves detallados por McGraw (touchpoints)

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes DOI: https://doi.org/10.33414/ajea.1640.2024
Texto Completo Publicado en AJEA – Full Text Published in AJEA
https://rtyc.utn.edu.ar/index.php/ajea/article/view/1628

En adición a las características mencionadas anteriormente que describen al desarrollo seguro, podemos encontrar los problemas de seguridad más importantes de las páginas web en el informe de 2021 del Top 10 de riesgos críticos en las aplicaciones web de OWASP.

1. Pérdida de Control de Acceso. Hace referencia a las debilidades detectadas en la implementación de controles de autenticación y autorización.
2. Fallas Criptográficas. Estas son fallas relacionadas con la criptografía, estas fallas frecuentemente conllevan a la exposición de datos confidenciales o al compromiso del sistema.
3. Inyección. Estos ataques ocurren cuando se envían datos que no son de confianza a un intérprete de código a través de la entrada de un formulario o algún otro envío de datos a una aplicación web.
4. Diseño Inseguro. Se necesita más modelado de amenazas, utilización más patrones y principios de diseño seguro y arquitecturas de referencia. Un diseño inseguro no puede ser corregido con una implementación debido a que los controles de seguridad necesarios nunca se crearon para defenderse de ataques específicos.
5. Configuración de Seguridad Incorrecta. Esto puede ocurrir al usar configuraciones por defecto o en mostrar errores en forma muy detallada.
6. Componentes Vulnerables y Desactualizados. Es sabido que los desarrolladores web utilizan componentes como bibliotecas y marcos para el desarrollo de código. Estos componentes son piezas de software que ayudan a los desarrolladores a evitar el trabajo redundante y a ofrecer la funcionalidad necesaria; pero a su vez algunos atacantes buscan vulnerabilidades dentro de estos componentes.
7. Fallas de Identificación y Autenticación: están relacionadas con fallas de identificación.
8. Fallas en la Integridad del Software y de los Datos, enfocándose en realizar hipótesis relacionadas con actualizaciones de software, datos críticos y pipelines de CI/CD sin verificar su integridad.
9. Fallas en el Registro y Monitoreo de la Seguridad. Esta categoría se amplía para incluir más tipos de fallas que son difíciles de probar y no están bien representadas en los datos de CVE/CVSS. Igualmente, las fallas incluidas en esta categoría pueden afectar directamente la visibilidad, las alertas de incidentes y los análisis forenses.
10. Falsificación de Solicitudes del Lado del Servidor (SSRF). Esta categoría representa el escenario en el que los miembros de la comunidad de seguridad nos dicen que esto es importante, a pesar de no ser visible en los datos en este momento.

Problemas de vulnerabilidades de los sistemas

Las organizaciones al momento de desarrollar sus aplicaciones web se ven obligadas a llevar a cabo un constante monitoreo de amenazas a lo largo del SDLC (Systems Development Life Cycle) según lo establecido por la norma ISO 27001. Existen diversos tipos de ataques a la que están constantemente expuestos y son muy comunes, en el cual si en estas no se tomaron sus debidas medidas o prevenciones pueden derivar en la pérdida o robo de la información. A continuación, se enumeran algunos tipos de ataques, entre ellos los más comunes a los que están expuestos las organizaciones son:

SQL Injection

Es uno de los tipos de ataque más comunes y recurrentes en el mundo, está catalogado como uno de los diez riesgos más críticos en la OWASP. La gran mayoría de aplicaciones web utilizan una base de datos para almacenar los distintos tipos de información a operar y que para acceder a ellos se utiliza un lenguaje de consulta estructurada (SQL) el cual lee, actualiza, agrega y elimina la información almacenada en la base de datos. Si esta parte del proceso falla o se realiza de forma insegura, la aplicación podría ser vulnerable a una inyección SQL (Stuttard et al, 2011).

Para el atacante que logre vulnerar estas solicitudes del servidor, le será posible obtener información crucial de la base de datos, ya sea leyendo, modificando los datos almacenados e incluso en el peor de los casos tomando el control del servidor.

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes DOI: https://doi.org/10.33414/ajea.1640.2024
Texto Completo Publicado en AJEA – Full Text Published in AJEA
https://rtyc.utn.edu.ar/index.php/ajea/article/view/1628

Algunas de las inyecciones más comunes son: SQL, NoSQL, Object-Relational Mapping (ORM)

Hoy en día las vulnerabilidades de las inyecciones SQL se han vuelto menos extendidas y más difíciles de detectar y explotar. Los errores de sintaxis del sistema pueden ser sutiles y provocar que sea muy difícil de distinguir en la categoría de vulnerabilidades que no representan una amenaza de seguridad.

Buffer overflows

El desbordamiento de buffer (Buffer overflows) surge cuando las aplicaciones web están escritas en código nativo o poseen alguna parte de ella. Si la aplicación no está libre de dicha vulnerabilidad, la aplicación copia los datos controlados por el usuario en un buffer de memoria el cual no es lo suficientemente grande para contenerlos, desbordándose y sobre escribiendo los datos del usuario.

Los atacantes pueden explotarlas ejecutando código arbitrario en el servidor.

XXS, cross-site scripting

Los Cross-Site Scripting (XSS) son un tipo de inyección en el cual se insertan scripts malintencionados en los sitios web confiables. Cuando un atacante se da cuenta de dicha vulnerabilidad, le puede enviar el script malicioso a un usuario que esta desprevenido y este ejecutarlo sin saberlo,haciendo que el script pueda acceder a las cookies, token de sesión u otra información guardada en el navegador.

Algunos tipos de XXS [5]: Reflected XSS (Type I), Stored XSS (Type II), DOM Based XSS (Type-0).

Mecanismos de mitigación

PENTESTING

El pentesting es una metodología realizada para descubrir vulnerabilidades y/o fallos de seguridad en un sistema informático, como puede ser una página web. Está diseñado para clasificar y determinar los alcances y las repercusiones de los fallos de seguridad, dando resultados para las empresas en cuanto a poder identificar la información o entornos a los cuales se podrían alcanzar en un ataque, además de evaluar la eficiencia de la defensa con la que cuentan (Vanegas 2019:2).

Las pruebas de penetración o pentesting son las prácticas llevadas a cabo a lo largo del proceso de desarrollo de software para validar y encontrar errores de vulnerabilidades en una aplicación web. Posee diferentes fases en la metodología DevSecOps, como se puede apreciar en la figura 2, la cual se llevará a cabo mezclando herramientas automatizadas y supervisado manualmente por un pentester. El pentesting es una parte importantísima para la organización ya que al implementarla en el proyecto te asegura cumplir con las normas ISO 27001.

Estas pruebas se implementan durante el desarrollo del código cuando aún no se lanzó una prueba de ejecución real en el servidor, luego cuando está ya está lista para lanzarse en el servidor y por último cuando está funcionando para el público.

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes DOI: https://doi.org/10.33414/ajea.1640.2024
Texto Completo Publicado en AJEA – Full Text Published in AJEA
https://rtyc.utn.edu.ar/index.php/ajea/article/view/1628

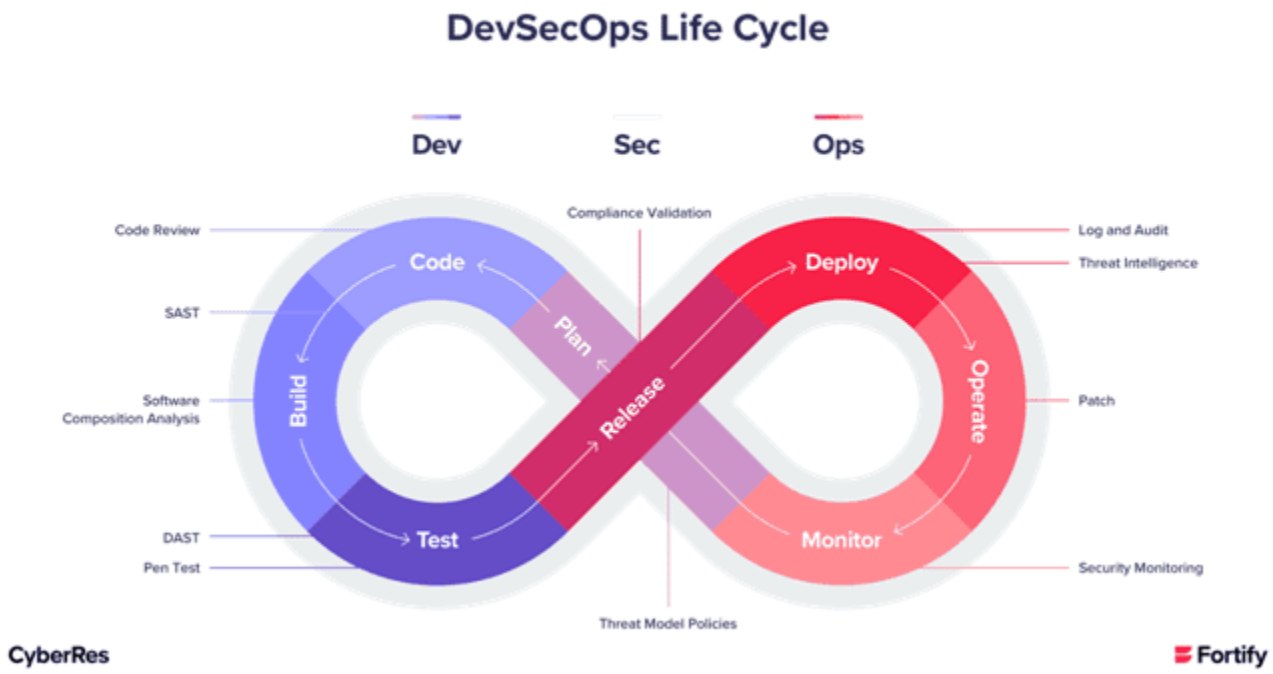

Figura 2 – Describe el ciclo de vida de DevSecOps

SAST

Las pruebas de seguridad estática (SAST) o también conocida como prueba de caja blanca, es una herramienta automatizada que escanea y revisa el código fuente, de adentro hacia afuera.

Esto ayuda a los desarrolladores a encontrar rápidamente las vulnerabilidades del sistema en una etapa temprana del desarrollo antes de integrar el software en un servidor de ejecución.

Ventajas:

- Garantiza la seguridad en las primeras etapas de desarrollo
- Fácil de integrar
- Es eficiente para el probador

Desventajas:

- Se necesita poseer el código fuente, por lo que no es una simulación realista de un ataque informático
- Altos números de falsos positivos
- Poca visibilidad en la lógica empresarial

DAST

Las pruebas de seguridad dinámica (DAST) o también conocida como prueba de caja negra, es una herramienta utilizada cuando no se tiene información previa acerca de la aplicación, como su seguridad, estructura, código fuente, entre otras, y está ya está en ejecución el en servidor.

Ventajas:

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes DOI: https://doi.org/10.33414/ajea.1640.2024
Texto Completo Publicado en AJEA – Full Text Published in AJEA
https://rtyc.utn.edu.ar/index.php/ajea/article/view/1628

- No requiere del código fuente para la prueba, por lo que es más realista
- Es el mismo escenario en el que se enfrentaría el atacante
- Monitoreo continuo

Desventajas:

- Se utiliza cuando ya existe un ambiente en ejecución
- Las vulnerabilidades son más costosas de corregir
- Falsos positivos
- Se puede demorar varios días en completar el escaneo
- Tiende a ser un proceso lento

IAST

Las pruebas de seguridad de aplicaciones interactivas (IAST) se clasificarían entre las pruebas de caja blanca y de caja negra, denominándola caja gris. Es una de las herramientas más nuevas y se utiliza cuando el código ya está en ejecución. Se obtiene acceso al código de la aplicación, a su flujo de datos y de control, analizándolo cuando ya está en el servidor en funcionamiento. En el momento en el cual detecte una vulnerabilidad, ésta será marcada con extrema exactitud, aunque a comparación con la herramienta sast, ésta no escanea el código por completo, sino que solamente la parte que se está ejecutando.

Prueba de penetración manual

Las implementaciones de las herramientas antes nombradas son muy buenas, pero no serían eficientes sin alguien por detrás revisando el código manualmente (Manico et al., 2014). Los pentester son personas que investigan exhaustivamente el código en busca de problemas, fallos de lógica y la verificación de problemas descubiertos por estas herramientas.

Los pentester son una parte importante para el SDLC ya que cumplirían con la norma ISO 27002 con la identificación y análisis de riesgos de la aplicación.

Las pruebas de penetración conllevan a la empresa no solo a mejorar la calidad de sus controles de seguridad y de protección de los datos sensibles, sino también a proteger su reputación, mantener la confianza del cliente y garantizar el cumplimiento normativo, entre otras cosas.

Controles ISO 27002

Algunos de los controles que se puede llevar a cabo:

- Política de seguridad: Esta debe incluir las directrices necesarias para el desarrollo seguro. La herramienta SAST ayuda a implementar la política de seguridad de la información, mientras que la DAST y IAST evalúa si esta se aplica efectivamente.
- Gestión de activos: con la codificación segura se puede identificar activos permitiendo una gestión mejor de los riesgos.
- Gestión de accesos: con la codificación segura se implementa los controles de accesos adecuados.
- Gestión de incidentes: las pruebas de penetración ayudan a evaluar que tan seguro son las aplicaciones ante los posibles ataques.
- Cumplimiento: Las herramientas nombradas anteriormente verifican el cumplimiento de los estándares de seguridad sobre la guía de buenas prácticas.

Sony sufrió un ataque de inyección SQL

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes DOI: https://doi.org/10.33414/ajea.1640.2024
Texto Completo Publicado en AJEA – Full Text Published in AJEA
https://rtyc.utn.edu.ar/index.php/ajea/article/view/1628

El grupo identificado como Lulz Security, utilizó un ataque simple de inyección SQL en la página web de la empresa, accediendo a información privilegiada de un millón de usuarios. Tal información era nombres de usuario, contraseñas y emails de los mismos.

El grupo atacante publicó un comunicado en donde exponían miles de cuentas de usuarios, donde además se criticaba a la empresa de no tener cifrada la información del usuario que esta estaba en formato texto, y que por otra parte era fácilmente legible.

La empresa Sony no dio muchos detalles al público más que un aviso de que estaban investigando sobre el ataque a su página web.

En este caso práctico se aprecia que hasta una de las mayores empresas a nivel internacional no está exenta de problemas de vulnerabilidad tan simples como sería un ataque SQL, perdiendo así información sensible. Este caso se pudo haber evitado si se tuvieran en cuenta las herramientas mencionadas anteriormente como realizar una prueba de seguridad manual o pentesting.

De acuerdo con el informe de Seguridad del 2022 realizado por Check Point Software, los ciberataques globales aumentaron un 38% en 2022 en comparación con 2021, lo que sugiere que es necesario tomar nuevas medidas de seguridad informacional, como las que fueron analizadas en este artículo, a la hora de lanzar cualquier sistema informático.

Conclusiones

Probar el software para la seguridad suele ser un proceso costoso y difícil si se deja para la etapa final del SDLC, por lo cual lo mejor sería presupuestar todo lo antes posible en la etapa más temprana del desarrollo. La mejor prevención seria combinando estas herramientas en diferentes etapas del SDLC, utilizando desde la etapa más temprana las herramientas de análisis estático de código (SAST), luego siguiendo con el análisis dinámico (DAST) y pruebas de seguridad de aplicaciones interactivas (IAST) en la etapa de ejecución del código en el servidor, teniendo a un pentester para la evaluación de la lógica misma y revisión manual del código.

Por otra parte, si no se implementó ninguna de estas herramientas a lo largo del SDLC, se pueden implementar con el producto terminado, pero los costos serán mayores y llevara bastante tiempo.

En conclusión, esto hace reflexionar que lleva tiempo la preparación de una prueba de seguridad concisa antes de la ejecución de cualquier sistema, pero aun así es la única forma de conseguir resultados duraderos.

Referencias bibliografías

Ariza Bonces, D. M. (2019). Ethical hacking: una estrategia de defensa proactiva.

Check Point Software Technologies Ltd. (2023) “Check Point Software’s 2023 Cyber Security Report”. Disponible en < https://pages.checkpoint.com/cyber-security-report-2023.html>

EasyDMARC Inc. (2022) “¿Qué son las pruebas de penetración de caja negra, caja gris y caja blanca?” Disponible en <https://easydmarc.com/blog/es/que-son-las-pruebas-de-penetracion-de-caja-negra-caja-gris-y-caja-blanca/>.

J. Manico y A. Detlefsen, Iron-Clad Java (2014). *Building Se-cure Web Applications (Oracle Press).* McGraw-Hill Education Group.

McGraw, G. (2004). Software security. *IEEE Security & Privacy*, *2*(2), 80-83.

North Team (2022) “¿Qué es sast?”. Disponible en <https://www.north-networks.com/que-es-sast/>.

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Libro de actas de resúmenes DOI: https://doi.org/10.33414/ajea.1640.2024
Texto Completo Publicado en AJEA – Full Text Published in AJEA
https://rtyc.utn.edu.ar/index.php/ajea/article/view/1628

OWASP Foundation (2021) “OWASP Top Ten”. Disponible en <https://owasp.org/www-project-top-ten/>.

OWASP Foundation (2021) “A03:2021 – inyección.”. Disponible en <https://owasp.org/Top10/es/A032021−Injection/>.

OWASP Foundation (2021) “Types of xss.”. Disponible en <https://owasp.org/www-community/TypesofCross-SiteScripting>.

Snyk Limited (2023) “Interactive Application Security Testing (IAST)”. Disponible en <https://snyk.io/learn/application-security/iast-interactive-application-security-testing/>.

Stuttard D. y Pinto M. (2011) *The web application hacker’s handbook: Finding and exploiting security flaws*. John Wiley & Sons.

Vanegas Romero, A. Y. (2019). Pentesting, ¿Porque es importante para las empresas?.